Web Dev Day 3: CSS3 Guide Part 1

dev.to/_bhupeshk_/web-dev-day-3-css3-guide-part-1-1jpp

Bhupesh Kumar
Posted on Mar 4, 2024

#css #webdev #programming #coding

What is CSS?

CSS, or **Cascading Style Sheets**, is a style sheet language used to describe the presentation of a document written in HTML (Hypertext Markup Language) or XML (eXtensible Markup Language). CSS defines how elements on a web page should be displayed, including their layout, colors, fonts, spacing, and other visual aspects.

Basic format of CSS:

The basic format of a CSS (Cascading Style Sheets) rule consists of a selector and a set of declarations. Declarations define the styling properties that should be applied to the selected elements. Here's the general structure:

```
selector {
    property1: value1;
    property2: value2;
    /* More properties and values */
}
```

Selector: Specifies which HTML elements the rule should apply to. It can be an element selector (e.g., body, h1, p), a class selector (e.g., .container), an ID selector (e.g., #header), or a combination of these.

Declaration Block: Enclosed in curly braces {}, contains one or more declarations. Each declaration includes a property, a colon :, and a value. Multiple declarations are separated by semicolons ;.

Property: Describes the styling aspect you want to change (e.g., color, font-size, margin).

Value: Specifies the value for the property (e.g., blue, 16px, 10px 20px).

Here's a simple example:

```
/* Selector: Apply styles to all paragraphs */
p {
    /* Declarations: Define styling properties */
    color: #333;
    font-size: 16px;
    margin-bottom: 10px;
}
```

How to include Styles:

There are several ways to include styles (CSS) in an HTML document. Here are three common methods:

1. External Stylesheet:
In this approach, you create a separate CSS file with the styling rules and link it to your HTML document using the `<link>` element. This is a recommended method for larger projects as it keeps the HTML and CSS separate, promoting maintainability and reusability.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Your Web Page</title>
</head>
<body>
    <!-- Your HTML content here -->
</body>
</html>
```

2. Internal Styles (within `<style>` tag):
You can include your CSS directly within the HTML file using the `<style>` tag in the document's `<head>`. This method is suitable for smaller projects or when you have specific styles for a single page.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        /* Your CSS rules here */
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f0f0f0;
        }

        h1 {
            color: #0066cc;
        }
    </style>
    <title>Your Web Page</title>
</head>
<body>
    <!-- Your HTML content here -->
</body>
</html>
```

3. Inline Styles:
You can also apply styles directly to individual HTML elements using the style attribute. This method is suitable for making quick style adjustments, but it's not recommended for extensive styling due to maintainability concerns.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Your Web Page</title>
</head>
<body>
    <!-- Your HTML content here with inline styles -->
    <h1 style="color: #0066cc;">Hello, World!</h1>
</body>
</html>
```

Color Property:

The color property in CSS is used to define the text color of an element. It sets the foreground color for the content of an element, such as text and hyperlinks. The value of the color property can be specified in various formats, including named colors, hexadecimal, RGB, or HSL values.

Here are some examples:

Named Colors:

```
p {
    color: red;
}
```

Hexadecimal Color:

```
h1 {
    color: #ff4500; /* Hexadecimal representation for orange color */
}
```

RGB Color:

```
h2 {
    color: rgb(0, 128, 0); /* RGB representation for green color */
}
```

HSL Color:

```
h3 {
    color: hsl(240, 100%, 50%); /* HSL representation for blue color */
}
```

RGBA Color (with alpha channel for transparency):

```
div {
    color: rgba(255, 0, 0, 0.5); /* Red color with 50% opacity */
}
```

CurrentColor:

The currentColor keyword is used to inherit the color property from the element's parent. This can be useful in certain scenarios, especially when working with dynamic styles or inheritance.

```
button {
    color: currentColor;
    border: 2px solid currentColor;
}
```

Background Color Property:

The background-color property in CSS is used to set the background color of an element. It defines the color behind the content, padding, and border areas of an element. The value of the background-color property can be specified using various color formats, including named colors, hexadecimal, RGB, or HSL values.

```
body {
    background-color: lightblue;
}
```

Color Systems:

Color systems in web development and design define how colors are represented and specified. There are several color systems commonly used in CSS:

- **Named Colors:** Examples: red, blue, green, etc. These are predefined color names that represent specific colors.
- **Hexadecimal (Hex) Color:** Example: #ff0000 (represents red). Hexadecimal values range from 00 to ff for each of the RGB (Red, Green, Blue) components.
- **RGB Color:** Example: rgb(255, 0, 0) (represents red). RGB uses three values (0 to 255) to represent the intensity of red, green, and blue.
- **HSL Color:** Example: hsl(0, 100%, 50%) (represents red). HSL (Hue, Saturation, Lightness) uses three values to represent the color in terms of its hue, saturation, and lightness.
- **RGBA Color:** Example: rgba(255, 0, 0, 0.5) (represents red with 50% opacity). Similar to RGB, but with an additional alpha channel for controlling opacity (transparency).
- **HSLA Color:** Example: hsla(0, 100%, 50%, 0.5) (represents red with 50% opacity). Similar to HSL but with an additional alpha channel for controlling opacity.

Number Systems:

Number systems are mathematical notation systems for expressing numbers in a consistent and standardized way. In the context of computing and programming, the most commonly used number systems are the decimal system (base-10), binary system (base-2), octal system (base-8), and hexadecimal system (base-16).

1. Decimal System (Base-10):
Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Each position represents a power of 10.
Example: 123 (1 * 10^2 + 2 * 10^1 + 3 * 10^0)

2. Binary System (Base-2):
Digits: 0, 1
Each position represents a power of 2.
Example: 1101 (1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0)

3. Octal System (Base-8):
Digits: 0, 1, 2, 3, 4, 5, 6, 7
Each position represents a power of 8.
Example: 53 (5 * 8^1 + 3 * 8^0)

4. Hexadecimal System (Base-16):
Digits: 0-9, A-F (where A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)
Each position represents a power of 16.
Example: 1A3 (1 * 16^2 + 10 * 16^1 + 3 * 16^0)

5. Roman Numerals:
Not used in computing but historically significant.
Symbols: I (1), V (5), X (10), L (50), C (100), D (500), M (1000)
Examples: III (3), XV (15), XC (90)

Text Properties:

In CSS, text properties are used to control the styling and presentation of text content within HTML elements. Here are some common text properties:

1. color:
Specifies the color of the text.
Example: `color: #333;`

2. font-family:
Defines the font or font stack used for the text.
Example: `font-family: 'Arial', sans-serif;`

3. font-size:
Sets the size of the text.
Example: `font-size: 16px;`

4. font-weight:
Specifies the thickness or boldness of the text.
Example: `font-weight: bold;`

5. font-style:
Sets the style of the font (normal, italic, or oblique).
Example: `font-style: italic;`

6. line-height:
Defines the height of a line of text.
Example: `line-height: 1.5;`

7. text-align:
Specifies the horizontal alignment of text.
Example: `text-align: center;`

8. text-decoration:
Sets the decoration of text (underline, overline, line-through).
Example: `text-decoration: underline;`

9. text-transform:
Controls the capitalization of text.
Example: `text-transform: uppercase;`

10. letter-spacing:
Adjusts the space between characters in the text.
Example: `letter-spacing: 2px;`

11. word-spacing:
Adjusts the space between words in the text.
Example: `word-spacing: 4px;`

12. white-space:
Controls how white space inside an element is handled.
Example: `white-space: nowrap;`

13. text-shadow:
Adds a shadow effect to the text.
Example: `text-shadow: 2px 2px 4px #000;`

14. overflow:
Specifies how text should behave when it overflows its container.
Example: `overflow: hidden;`

15. text-overflow:
Controls how text is truncated when it overflows its container.
Example: `text-overflow: ellipsis;`

Text Align Property:

The text-align property in CSS is used to set the horizontal alignment of text content within a block-level element. It determines whether the text should be aligned to the left, right, center, or justified within its container.

Here are the possible values for the text-align property:

1. left:
Aligns the text to the left edge of the containing element.
Example: `text-align: left;`

2. right:
Aligns the text to the right edge of the containing element.
Example: `text-align: right;`

3. center:
Centers the text horizontally within the containing element.
Example: `text-align: center;`

4. justify:
Stretches the lines of text so that each line has equal width, creating a justified effect.
Example: `text-align: justify;`

5. initial:
Sets the text-align property to its default value.
Example: `text-align: initial;`

6. inherit:
Inherits the text-align property from its parent element.
Example: `text-align: inherit;`

Example Usage:

```
/* Center-align text within a div */
div {
    text-align: center;
}

/* Right-align text within paragraphs */
p {
    text-align: right;
}

/* Justify text within an article */
article {
    text-align: justify;
}
```

Font Weight:

The **font-weight** property in CSS is used to specify the thickness or boldness of the characters in text. It can take various values ranging from numeric values to named values.

Here are some common values for the font-weight property:

1. Numeric Values:
Values from 100 to 900, with 100 being the lightest and 900 being the boldest.
Example: `font-weight: 700;` (bold)

2. Named Values:
Common names include normal (equivalent to 400), bold (equivalent to 700), and bolder/lighter (relative to the inherited font weight).
Example: `font-weight: bold;`

Example Usage:

```
/* Set font weight to bold */
h1 {
    font-weight: bold;
}

/* Set font weight to 600 (semi-bold) */
p {
    font-weight: 600;
}

/* Inherit font weight from the parent element */
span {
    font-weight: inherit;
}
```

Text Decoration:

The **text-decoration** property in CSS is used to set or remove decorations on text. Decorations include underline, overline, and line-through. This property is commonly used to enhance the visual appearance of links or to add stylistic effects to text.

Here are the possible values for the text-decoration property:

1. none:
Removes any text decoration (default value).
Example: `text-decoration: none;`

2. underline:
Adds an underline to the text.
Example: `text-decoration: underline;`

3. overline:
Adds a line above the text.
Example: `text-decoration: overline;`

4. line-through:
Adds a line through the middle of the text.
Example: `text-decoration: line-through;`

5. underline overline:
Adds both underline and overline to the text.
Example: `text-decoration: underline overline;`

6. underline line-through:
Adds both underline and line-through to the text.
Example: `text-decoration: underline line-through;`

Example Usage:

```
/* Remove text decoration for links */
a {
    text-decoration: none;
}

/* Add underline to the heading */
h1 {
    text-decoration: underline;
}

/* Add line-through to a paragraph */
p {
    text-decoration: line-through;
}

/* Add both underline and overline to a span */
span {
    text-decoration: underline overline;
}
```

Line Height:

The **line-height** property in CSS is used to set the height of a line of text within a block-level element. It specifies the amount of space above and below the inline content (such as text) in the element. The value of line-height can be expressed as a unitless number, a percentage, or in length units.

Here are different ways to use the line-height property:

1. Unitless Number:
Represents a multiplier relative to the font size of the element.
Example: `line-height: 1.5;` (1.5 times the font size)

2. Percentage:
Represents a percentage of the font size of the element.
Example: `line-height: 150%;` (150% of the font size)

3. Length Units:
Can be specified in absolute length units (px, em, rem, etc.).
Example: `line-height: 24px;` (24 pixels)

Example Usage:

```
/* Set line height as a unitless number */
p {
    line-height: 1.5;
}

/* Set line height as a percentage */
h1 {
    line-height: 120%;
}

/* Set line height using length units */
div {
    line-height: 1.2em;
}
```

Letter Spacing:

The **letter-spacing** property in CSS is used to control the amount of space between characters (letters) in a block of text. It can be used to increase or decrease the space between characters, affecting the overall spacing and readability of the text.

Here are different ways to use the letter-spacing property:

1. Normal (Default):
Default spacing between characters.
Example: `letter-spacing: normal;`

2. Length Units:
Specifies the space between characters in absolute length units (px, em, rem, etc.).
Example: `letter-spacing: 2px;` (2 pixels of space between characters)

Example Usage:

```
/* Use normal letter spacing for paragraphs */
p {
    letter-spacing: normal;
}

/* Increase letter spacing for a heading */
h1 {
    letter-spacing: 1.5px;
}

/* Decrease letter spacing for a navigation menu */
nav a {
    letter-spacing: -0.5px;
}
```

Font Size Units in CSS:

In CSS, font sizes can be specified using various units. Each unit has its own characteristics and use cases. Here are some common font size units in CSS:

1. Pixels (px):
Absolute unit, representing a fixed-size length.
Example: `font-size: 16px;`

2. Ems (em):
Relative unit, representing the current font size of the element.
Example: `font-size: 1.5em;` (1.5 times the current font size)

3. Rems (rem):
Relative unit, similar to em, but relative to the root (html) font size.
Example: `font-size: 1.2rem;` (1.2 times the root font size)

4. Percentages (%):
Relative unit, representing a percentage of the parent element's font size.
Example: `font-size: 120%;` (120% of the parent element's font size)

5. Viewport Width (vw):
Relative unit, representing a percentage of the viewport width.
Example: `font-size: 5vw;` (5% of the viewport width)

6. Viewport Height (vh):
Relative unit, representing a percentage of the viewport height.
Example: `font-size: 8vh;` (8% of the viewport height)

7. Viewport Minimum (vmin):
Relative unit, representing the smaller of vw or vh.
Example: `font-size: 6vmin;` (6% of the smaller of viewport width or height)

8. Viewport Maximum (vmax):
Relative unit, representing the larger of vw or vh.
Example: `font-size: 10vmax;` (10% of the larger of viewport width or height)

Example Usage:

```
/* Set font size in pixels */
body {
    font-size: 16px;
}

/* Set font size in ems */
h1 {
    font-size: 2em;
}

/* Set font size in rems */
p {
    font-size: 1.2rem;
}

/* Set font size in percentages */
div {
    font-size: 120%;
}

/* Set font size using viewport units */
article {
    font-size: 5vw;
}
```

Font Family:

The **font-family** property in CSS is used to define the typeface or font family that should be applied to the text content of an element. Font families are groups of typefaces sharing a similar design, and the font-family property allows you to specify a prioritized list of font family names or generic family names.

Here are different ways to use the font-family property:

1. Specific Font Family:
Specifies the exact font family to be used.
Example: `font-family: 'Arial', sans-serif;`

2. Generic Font Family:
Specifies a generic font family, allowing the browser to choose an appropriate font.
Example: `font-family: serif;`

3. Multiple Font Families:
Specifies a prioritized list of font families. If the browser doesn't have the first font, it will try the next one in the list.
Example: `font-family: 'Helvetica', 'Arial', sans-serif;`

Example Usage:

```
/* Use a specific font family */
body {
    font-family: 'Arial', sans-serif;
}

/* Use a generic font family */
h1 {
    font-family: serif;
}

/* Specify multiple font families */
p {
    font-family: 'Helvetica', 'Arial', sans-serif;
}
```

CSS Fonts: https://www.cssfontstack.com/

Text Transform:

The **text-transform** property in CSS is used to control the capitalization of text. It allows you to specify whether the text should be displayed in uppercase, lowercase, or with each word capitalized.

Here are the possible values for the text-transform property:

1. none:
Default value. Text is displayed as entered.

2. uppercase:
Transforms all characters to uppercase.
Example: `text-transform: uppercase;`

3. lowercase:
Transforms all characters to lowercase.
Example: `text-transform: lowercase;`

4. capitalize:
Transforms the first character of each word to uppercase.
Example: `text-transform: capitalize;`

Example Usage:

```
/* Keep text as entered */
p {
    text-transform: none;
}

/* Display text in uppercase */
h1 {
    text-transform: uppercase;
}

/* Display text in lowercase */
span {
    text-transform: lowercase;
}

/* Capitalize the first character of each word */
div {
    text-transform: capitalize;
}
```

Selectors:

Selectors in CSS are patterns that are used to select and style HTML elements on a webpage. They define the elements to which a set of CSS rules should be applied. Selectors can be simple, targeting specific elements, or complex, combining multiple conditions to select elements more precisely.

Here are some common types of CSS selectors:

1. Type/Element Selector:
Selects all instances of a given HTML element type.
Example: p selects all `<p>` elements.

2. Class Selector:
Selects all elements with a specific class attribute.
Example: `.highlight` selects all elements with `class="highlight"`.

3. ID Selector:
Selects a single element with a specific id attribute.
Example: `#header` selects the element with `id="header"`.

4. Descendant Selector:
Selects an element that is a descendant of another specified element.
Example: `article p` selects all `<p>` elements that are descendants of `<article>` elements.

5. Child Selector:
Selects an element that is a direct child of another specified element.
Example: `ul > li` selects all `<li>` elements that are direct children of `<ul>` elements.

6. Attribute Selector:
Selects elements based on the presence or value of their attributes.
Example: `[type="text"]` selects all elements with `type="text"`.

7. Pseudo-class Selector:
Selects elements based on their state or position.
Example: `a:hover` selects `<a>` elements when hovered.

8. Pseudo-element Selector:
Selects a part of an element, such as the first line or first letter.
Example: `p::first-line` selects the first line of all `<p>` elements.

9. Grouping Selector:
Groups multiple selectors together to apply the same styles.
Example: `h1, h2, h3` selects all `<h1>`, `<h2>`, and `<h3>` elements.

10. Universal Selector:
The universal selector * in CSS selects all elements on a webpage. It matches any element type and is often used to apply styles globally.

Sibling Combinator:
The sibling combinator (+) in CSS is used to select an element that is immediately preceded by a specified element. It targets the next sibling element that shares the same parent.

Example Usage:

```
/* Type/Element Selector */
p {
    color: blue;
}

/* Class Selector */
.highlight {
    background-color: yellow;
}

/* ID Selector */
#header {
    font-size: 24px;
}

/* Descendant Selector */
article p {
    font-style: italic;
}

/* Child Selector */
ul > li {
    list-style-type: square;
}

/* Attribute Selector */
input[type="submit"] {
    background-color: green;
}

/* Pseudo-class Selector */
a:hover {
    text-decoration: underline;
}

/* Grouping Selector */
h1, h2, h3 {
    font-family: 'Arial', sans-serif;
}

/* Universal Selector */
* {
    border: 1px solid #ccc;
}

/* Sibling Combinator */
p + h2 {
    color: red;
}
```

Cascading:

Cascading, in the context of Cascading Style Sheets (CSS), refers to the process by which multiple style rules are applied to an HTML element, and the browser determines the final styles that should be used for that element. The "cascading" nature of CSS comes from the way different stylesheets and style rules interact and override each other.

Specificity:

Specificity in CSS is a measure of how specific a selector is in targeting HTML elements. It determines which styles should be applied when there are conflicting styles for the same element. Specificity is crucial in the cascading process of determining the final styles for an element.

Specificity Hierarchy:

Inline Styles:

- Highest specificity.
- Applied directly to an element using the style attribute.
- Example: `<p style="color: red;">This is red text.</p>`

ID Selectors:

- More specific than class or element selectors.
- Represented by a hash (#) followed by the ID value.
- Example: #header { color: blue; }

Class and Attribute Selectors:

- Less specific than ID selectors.
- Represented by a period (.) for class selectors or brackets ([]) for attribute selectors.
- Example: `.highlight { font-weight: bold; } or [type="text"] { border: 1px solid #ccc; }`

Element Selectors:

- Least specific.
- Target elements based on their type.
- Example: `p { font-size: 16px; }`

Specificity Calculation:

Specificity is represented as a four-part value, often written as a, b, c, d, where each part corresponds to the specificity of inline styles, ID selectors, class/attribute selectors, and element selectors, respectively.

- Example: 0, 1, 2, 3 means one ID selector is present.
- Example: 0, 0, 1, 3 means one class or attribute selector is present.

Rule for Comparing Specificity:

When ***two conflicting styles*** are encountered, the style with the ***higher specificity wins***. If ***specificity is equal***, the ***style that appears later in the stylesheet takes precedence***.

!important:

The **!important** declaration in CSS is used to give a style rule higher specificity and priority, making it override other conflicting styles even if they have higher specificity or appear later in the stylesheet. It should be used sparingly, as it can make styles harder to maintain and debug due to its ability to override normal cascading rules.

Here's an example of using !important:

```
/* This style will be overridden */
p {
    color: blue;
}

/* This style will override the previous one due to !important */
p {
    color: red !important;
}
```

Inheritance:

Inheritance in CSS refers to the mechanism by which certain properties of a parent element are passed down to its child elements. This allows styles to be applied more efficiently, as properties set on a parent can be inherited by its descendants, reducing the need to explicitly apply styles to every individual element.

How Inheritance Works:

Inherited Properties:

Some CSS properties are inherited by default. Common examples include font-related properties (like `font-family`, `font-size`, `font-weight`), text-related properties (like `color`, `line-height`), and others.

Non-Inherited Properties:

Some properties are not inherited by default. These include properties like `border`, `margin`, `padding`, and others. Child elements don't inherit these values from their parent elements.

The inherit Keyword:

The `inherit` keyword can be explicitly used to force inheritance for a specific property. For example, `color: inherit;` will make the text color of a child element inherit the color from its parent.

Example:

```
/* Parent Styles */
body {
    font-family: 'Arial', sans-serif;
    font-size: 16px;
    color: #333;
}

/* Child Styles */
p {
    /* Inherits font-family and font-size from the body, but color is overridden
*/
    color: blue;
}

div {
    /* Inherits font-family, font-size, and color from the body */
    background-color: lightgray;
}

/* Explicitly force inheritance for the border color */
.border-inherit {
    border: 1px solid;
    border-color: inherit;
}
```

Coolors: https://coolors.co/

Canva: https://www.canva.com/colors/color-palette-generator/

Mycolor: https://mycolor.space/

Box Model:

The box model in CSS describes how elements are laid out on a webpage by defining the properties of the rectangular boxes that surround each element. Each HTML element is considered a box, and the box model comprises several components:

Content:
The actual content of the box, where text and images appear.

Padding:
The transparent space around the content inside the box.
Padding is defined using properties like `padding-top`, `padding-right`, `padding-bottom`, and `padding-left`.

Border:
The border surrounding the padding.
The border can have properties such as `border-width`, `border-style`, and `border-color`.

Margin:
The transparent space outside the border.
Margins separate one box from another.
Margin properties include `margin-top`, `margin-right`, `margin-bottom`, and `margin-left`.

Visual Representation of the Box Model:

```
+------------------------+
|       Margin           |
| +--------------------+ |
| |      Border        | |
| | +----------------+ | |
| | |    Padding     | | |
| | | +------------+ | | |
| | | |  Content   | | | |
| | | +------------+ | | |
| | +----------------+ | |
| +--------------------+ |
+------------------------+
```

Example:

```
/* Define styles for a box */
.box {
    width: 200px;
    height: 100px;
    padding: 20px;
    border: 2px solid #333;
    margin: 10px;
}
```

Display Property:

The display property in CSS is used to define how an HTML element should be rendered on the web page. Two common values for the display property are inline and block, each affecting the layout and behavior of elements in different ways:

1. Inline:
Elements with display: inline; are rendered inline with the text.
They do not start on a new line and only take up as much width as necessary.
Examples: `<span>`, `<a>`, `<strong>`

Example:

```
span {
    display: inline;
}
```

2. Block:
Elements with display: block; start on a new line and stretch the full width available. They create a "block" or a rectangular box.
Examples: `<div>`, `<p>`, `<h1>`

Example:

```
div {
    display: block;
}
```

3. Inline-Block:
The `display: inline-block;` in CSS is a property value that combines the characteristics of both inline and block-level elements. Here's a concise overview:

Inline Flow: Flows inline with text.
Block-Level Properties: Allows setting width, height, margins, and padding.
No Line Breaks: Does not start on a new line by default.

Inline vs Block Comparison:

Line Breaks:

Inline: Does not start on a new line; flows with the surrounding text.
Block: Starts on a new line; creates a block-level element.

Width:

Inline: Takes up only as much width as necessary.
Block: Stretches the full available width by default.

Height:

Inline: Height is determined by the content.
Block: Height is determined by the content or explicitly set.

Margin and Padding:

Inline: Horizontal margins and paddings can be applied, but vertical adjustments (top and bottom) may not work as expected.
Block: Accepts both horizontal and vertical margins and padding.

Examples: